# Exact Acceleration of Linear Object Detectors

Charles Dubout
François Fleuret

Idiap Research Institute

9 October 2012

# Plan

## Architecture of a modern linear object detector

- The sliding window technique
- HOG and linear SVM
- HOG feature planes

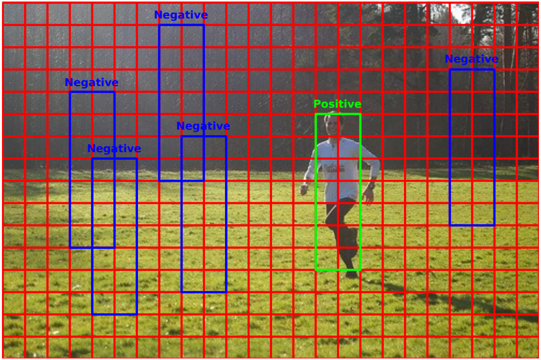
## Deformable part-based models (DPM)

- DPM use a lot of filters
- Challenge

## Our contribution

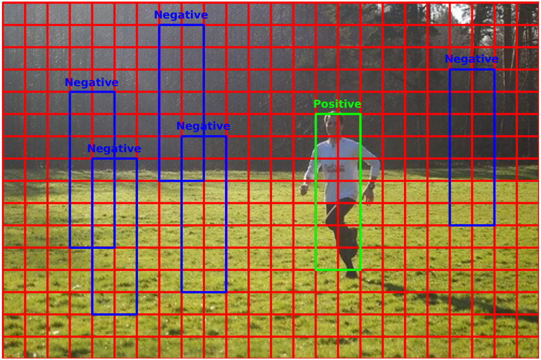
- Standard and Fourier convolution processes
- Patchworks of pyramid scales
- Cache violations
- Results

# The sliding window technique



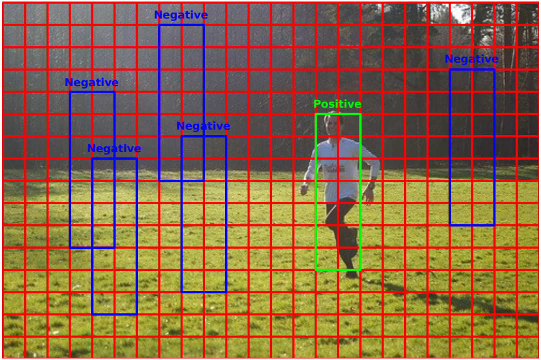

- Transforms a detection problem into a binary classification one

# The sliding window technique




- Transforms a detection problem into a binary classification one
- Applies a binary classifier at every image position and scale

# The sliding window technique

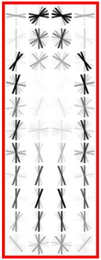



- Transforms a detection problem into a binary classification one
- Applies a binary classifier at every image position and scale
- Similar to sweeping the detection window across the whole image

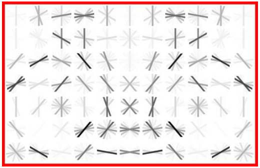
# HOG* and linear SVM

*Dalal & Triggs '05

Pedestrian template

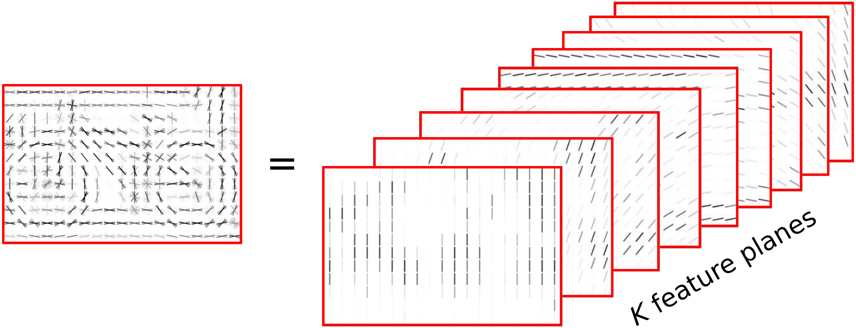


Bicycle template



Objects are image positions on the HOG grid: $score_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$, where $\mathbf{x}$ is the vector of features extracted from the subwindow at the position of interest of size same as $\mathbf{w}$.

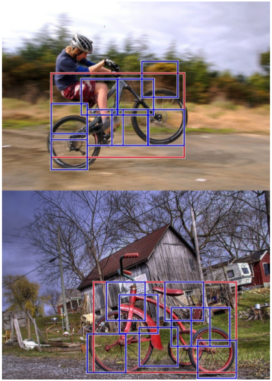
# HOG feature planes

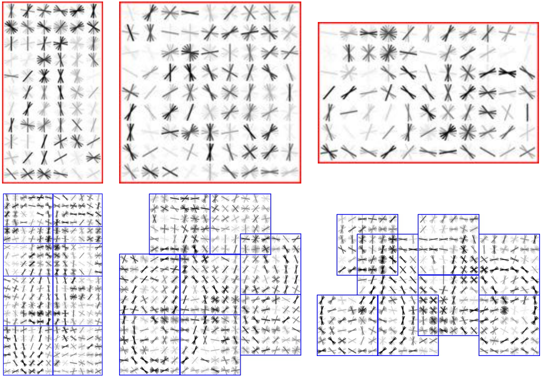



The HOG features can be seen as organized in planes, containing distinct features from each grid cell.

# DPM* use a lot of filters

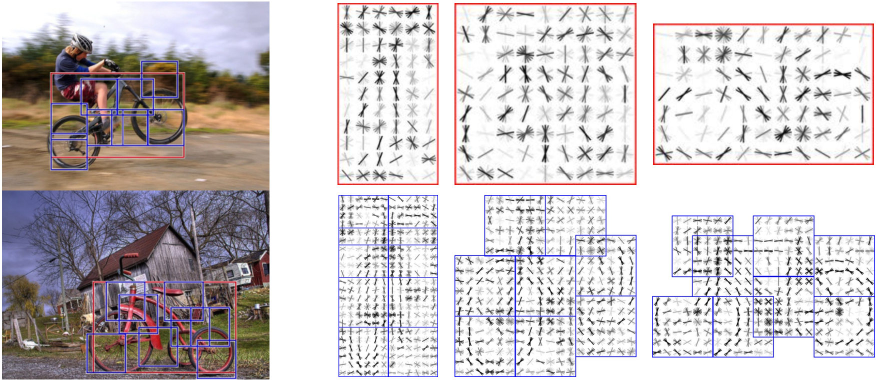
*Felzenszwalb & al. ’08





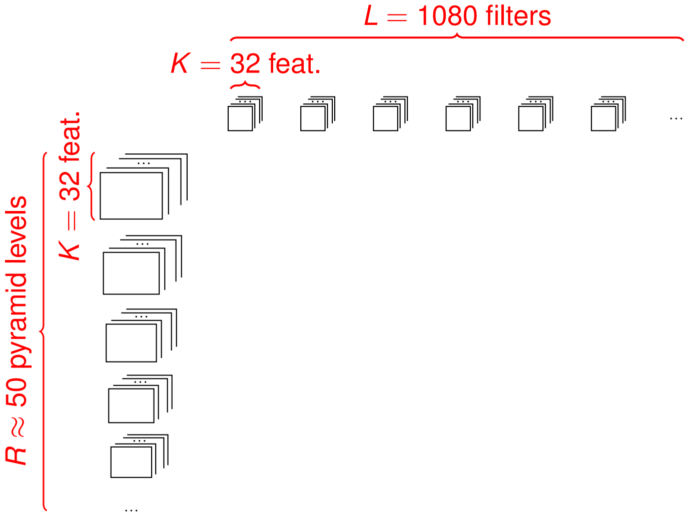
# DPM* use a lot of filters

*Felzenszwalb & al. '08

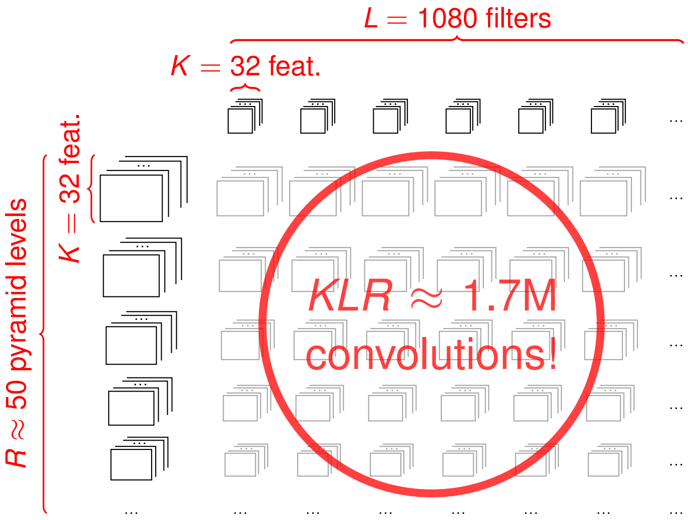


Typical numbers of filters used on the Pascal challenge:
20 classes × 6 mixtures × 9 parts = 1080 linear filters!

# Challenge

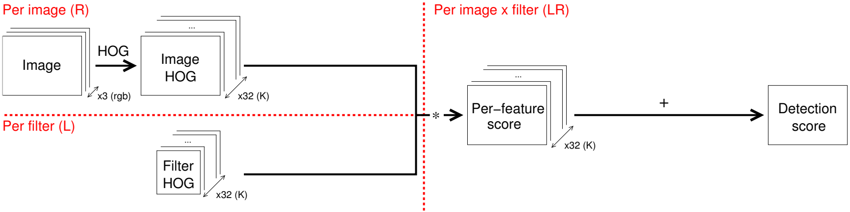


# Challenge

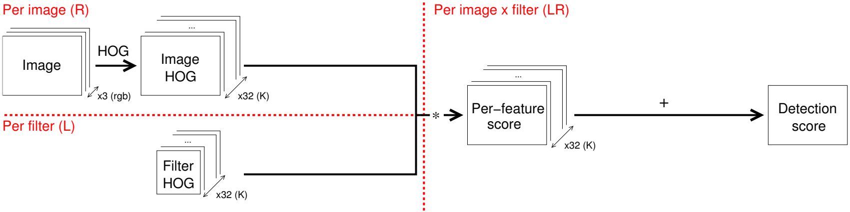


$L = 1080$ filters
$K = 32$ feat.
$K = 32$ feat.
$R \approx 50$ pyramid levels
$KLR \approx 1.7$M convolutions!

# Standard convolution process



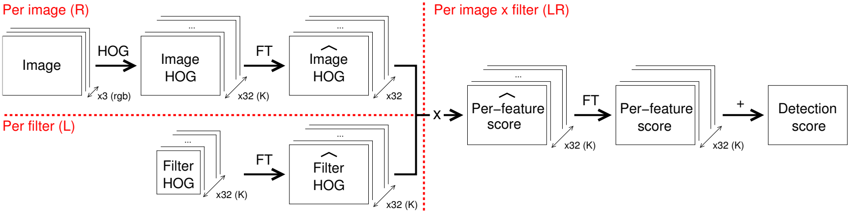
# Standard convolution process




The computational cost to convolve a HOG image of size $M \times N$ with $L$ filters of size $P \times Q$ across $K$ features is:

$$C_{\text{std}} = \mathcal{O}(KLMNPQ)$$

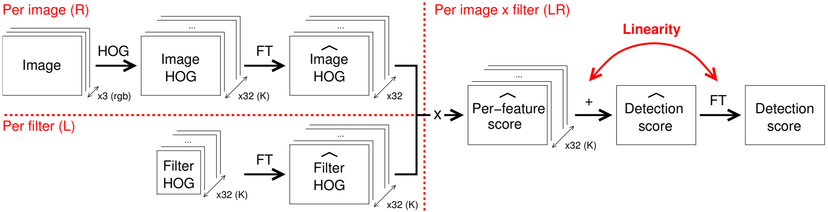
# Fourier based convolutions




The computational cost to convolve a HOG image of size $M \times N$ with $L$ filters of size $P \times Q$ across $K$ features is:

$$C_{\text{FFT}} = \underbrace{\mathcal{O}(KMN \log MN)}_{\text{Forward FFTs}} + \underbrace{\mathcal{O}(KLMN)}_{\text{Multiplications}} + \underbrace{\mathcal{O}(KLMN \log MN)}_{\text{Inverse FFTs}}$$

# Fourier based convolutions




The computational cost to convolve a HOG image of size $M \times N$ with $L$ filters of size $P \times Q$ across $K$ features is:

$$C_{\text{opt}} = \underbrace{\mathcal{O}(KMN \log MN)}_{\text{Forward FFTs}} + \underbrace{\mathcal{O}(KLMN)}_{\text{Multiplications}} + \underbrace{\mathcal{O}(\not{K}LMN \log MN)}_{\text{Inverse FFTs}}$$

$$\approx \mathcal{O}(KLMN)$$

# Lets plug in typical numbers

- $K = 32$ (number of HOG features)
- $L = 54$ (number of filters)
- $M \times N = 64 \times 64$ (size of the pyramid level)
- $P \times Q = 6 \times 6$ (size of the filters)

## Lets plug in typical numbers

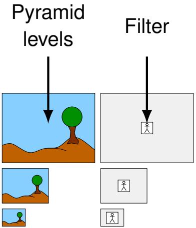
- $K = 32$ (number of HOG features)
- $L = 54$ (number of filters)
- $M \times N = 64 \times 64$ (size of the pyramid level)
- $P \times Q = 6 \times 6$ (size of the filters)

$$
\begin{aligned}
C_{\text{std}} &\approx 2KLMNPQ && \approx 490 \text{ MFlop} \\
C_{\text{FFT}} &\approx 3KLMN + 2.5(K + KL)MN \log_2 MN && \approx 230 \text{ MFlop} \\
C_{\text{opt}} &\approx 4KLMN + 2.5(K + L)MN \log_2 MN && \approx 37 \text{ MFlop}
\end{aligned}
$$

A gain by a factor 13 compared to the standard process, and 6 compared to the standard Fourier one!

# Patchworks of pyramid scales

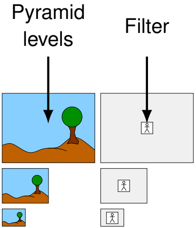
To use the FFT the image and the filter need to be of the same size.

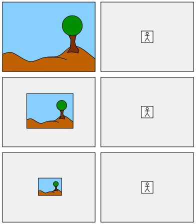



Memory inefficient

# Patchworks of pyramid scales

To use the FFT the image and the filter need to be of the same size.

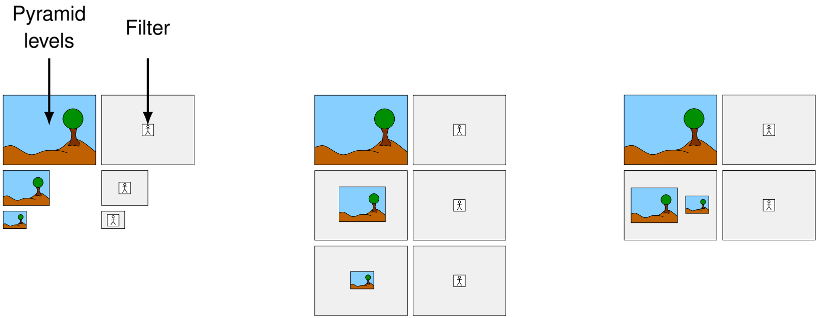


Memory inefficient    Computationally inefficient

# Patchworks of pyramid scales

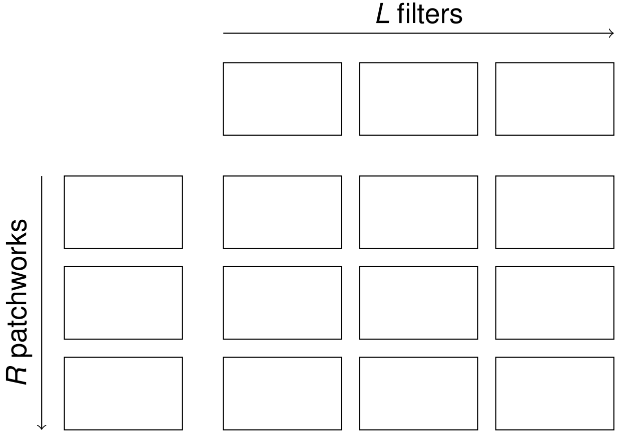
To use the FFT the image and the filter need to be of the same size.




Memory inefficient    Computationally inefficient    Best of both worlds

# Cache violations

## Naive strategy



# Cache violations

## Naive strategy


Read 2 into cache

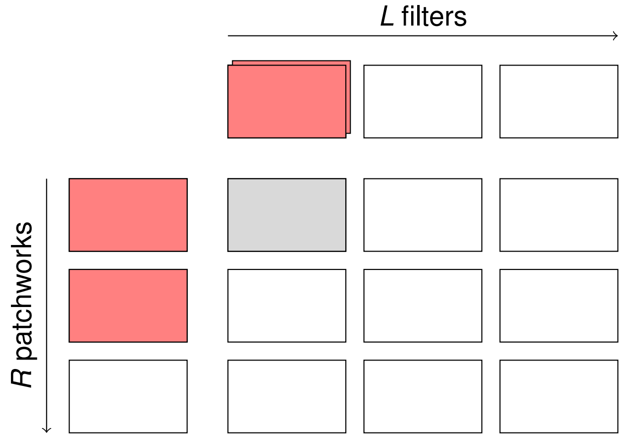
# Cache violations

## Naive strategy


Read 2 into cache $\Rightarrow$ compute 1.

# Cache violations

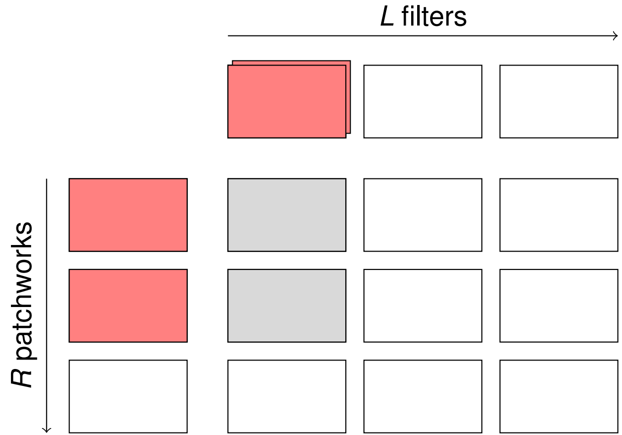
## Naive strategy




Read 2 into cache ⇒ compute 1.

# Cache violations

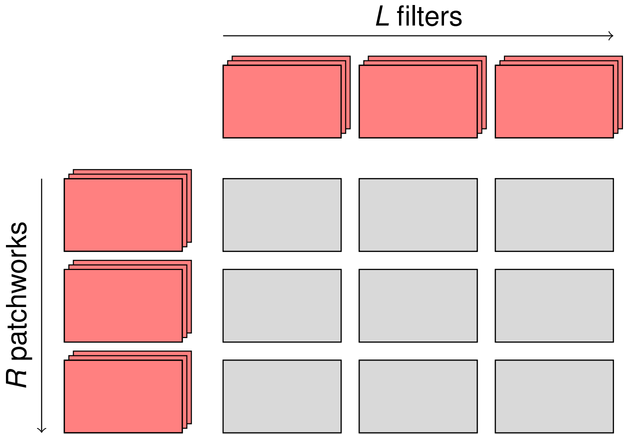
## Naive strategy




Read 2 into cache ⇒ compute 1.

# Cache violations

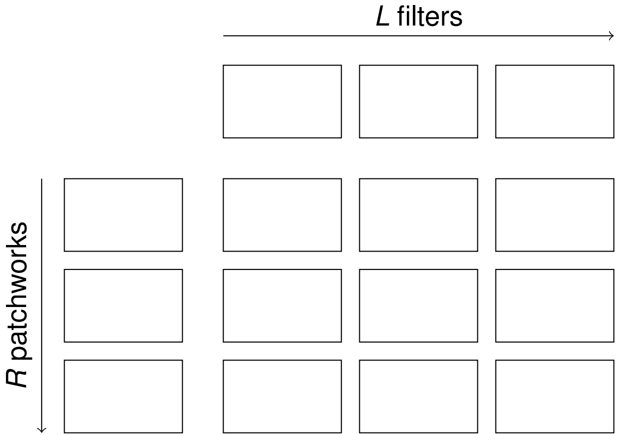
## Naive strategy




Read $2LR$ into cache $\Rightarrow$ compute $LR$.

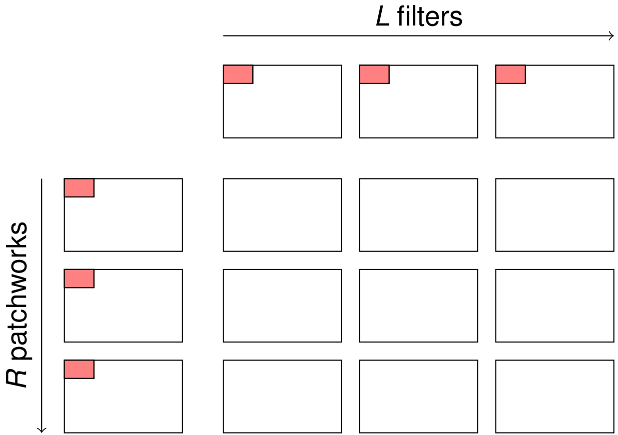
# Cache violations

## Fragment strategy



# Cache violations

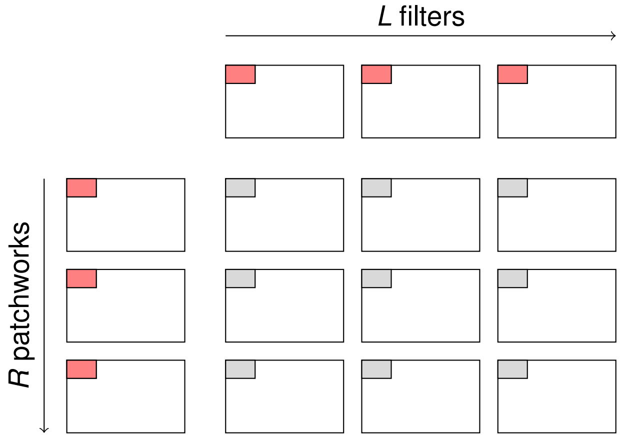
## Fragment strategy




Read $(L + R)\frac{\epsilon}{L+R} = \epsilon$ into cache

# Cache violations

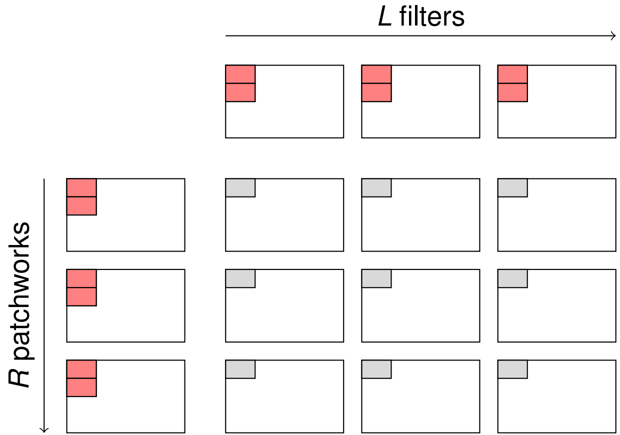
## Fragment strategy




Read $(L+R)\frac{\epsilon}{L+R} = \epsilon$ into cache $\Rightarrow$ compute $LR\frac{\epsilon}{L+R}$.

# Cache violations

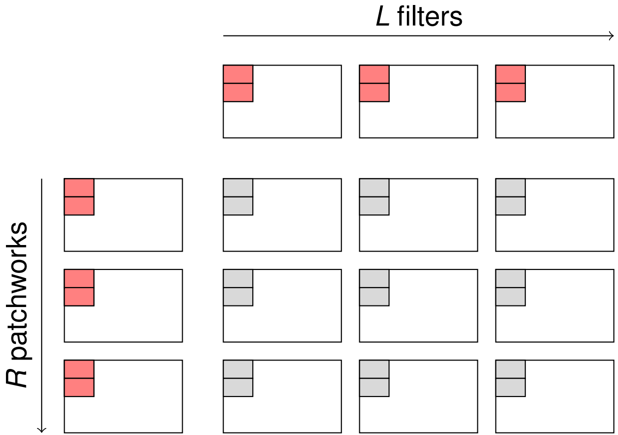
## Fragment strategy




Read $(L+R)\frac{\epsilon}{L+R} = \epsilon$ into cache $\Rightarrow$ compute $LR\frac{\epsilon}{L+R}$.

# Cache violations

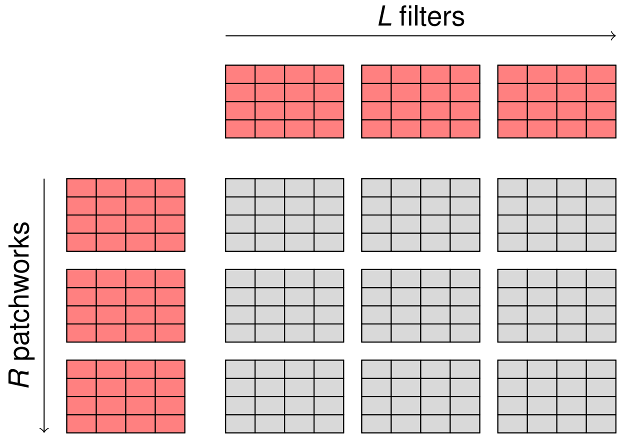
## Fragment strategy




Read $(L+R)\frac{\epsilon}{L+R} = \epsilon$ into cache $\Rightarrow$ compute $LR\frac{\epsilon}{L+R}$.

# Cache violations

## Fragment strategy




Read $L + R$ into cache $\Rightarrow$ compute $LR$.

# Results

Table : Pascal VOC 2007 challenge convolution time and speedup

| | **aero** | **bike** | **bird** | **boat** | **bottle** | **bus** | **car** | **cat** | **chair** | **cow** | **table** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **V4 (ms)** | 409 | 437 | 403 | 414 | 366 | 439 | 352 | 432 | 417 | 429 | 450 |
| **Ours (ms)** | 55 | 56 | 53 | 56 | 57 | 56 | 54 | 56 | 56 | 57 | 57 |
| **Speedup (x)** | 7.4 | 7.8 | 7.6 | 7.4 | 6.4 | 7.9 | 6.5 | 7.7 | 7.5 | 7.5 | 8.0 |

| | **dog** | **horse** | **mbike** | **person** | **plant** | **sheep** | **sofa** | **train** | **tv** | **mean** |
|---|---|---|---|---|---|---|---|---|---|---|
| **V4 (ms)** | 445 | 439 | 429 | 379 | 358 | 351 | 425 | 458 | 433 | **413** |
| **Ours (ms)** | 57 | 59 | 57 | 54 | 54 | 55 | 57 | 58 | 55 | **56** |
| **Speedup (x)** | 7.8 | 7.5 | 7.6 | 7.0 | 6.6 | 6.4 | 7.4 | 7.9 | 7.9 | **7.4** |

# Results

Table : Pascal VOC 2007 challenge convolution time and speedup

| | **aero** | **bike** | **bird** | **boat** | **bottle** | **bus** | **car** | **cat** | **chair** | **cow** | **table** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **V4 (ms)** | 409 | 437 | 403 | 414 | 366 | 439 | 352 | 432 | 417 | 429 | 450 |
| **Ours (ms)** | 55 | 56 | 53 | 56 | 57 | 56 | 54 | 56 | 56 | 57 | 57 |
| **Speedup (x)** | 7.4 | 7.8 | 7.6 | 7.4 | 6.4 | 7.9 | 6.5 | 7.7 | 7.5 | 7.5 | 8.0 |

| | **dog** | **horse** | **mbike** | **person** | **plant** | **sheep** | **sofa** | **train** | **tv** | **mean** |
|---|---|---|---|---|---|---|---|---|---|---|
| **V4 (ms)** | 445 | 439 | 429 | 379 | 358 | 351 | 425 | 458 | 433 | **413** |
| **Ours (ms)** | 57 | 59 | 57 | 54 | 54 | 55 | 57 | 58 | 55 | **56** |
| **Speedup (x)** | 7.8 | 7.5 | 7.6 | 7.0 | 6.6 | 6.4 | 7.4 | 7.9 | 7.9 | **7.4** |

- Error rate: identical to the baseline (32.3% AP)

# Results

Table : Pascal VOC 2007 challenge convolution time and speedup

| | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **V4 (ms)** | 409 | 437 | 403 | 414 | 366 | 439 | 352 | 432 | 417 | 429 | 450 |
| **Ours (ms)** | 55 | 56 | 53 | 56 | 57 | 56 | 54 | 56 | 56 | 57 | 57 |
| **Speedup (x)** | 7.4 | 7.8 | 7.6 | 7.4 | 6.4 | 7.9 | 6.5 | 7.7 | 7.5 | 7.5 | 8.0 |

| | dog | horse | mbike | person | plant | sheep | sofa | train | tv | mean |
|---|---|---|---|---|---|---|---|---|---|---|
| **V4 (ms)** | 445 | 439 | 429 | 379 | 358 | 351 | 425 | 458 | 433 | **413** |
| **Ours (ms)** | 57 | 59 | 57 | 54 | 54 | 55 | 57 | 58 | 55 | **56** |
| **Speedup (x)** | 7.8 | 7.5 | 7.6 | 7.0 | 6.6 | 6.4 | 7.4 | 7.9 | 7.9 | **7.4** |

- Error rate: identical to the baseline (32.3% AP)
- Numerical accuracy: better than the baseline ($1.8 \cdot 10^{-8}$ vs. $2.4 \cdot 10^{-8}$ MAE)

# Conclusion

- Part-based models obtain state-of-the-art performance at the price of a huge number of convolutions
- The FT is linear, enabling one to do the addition of the convolutions across feature planes in Fourier space
- The computational cost becomes invariant to the filters' sizes, resulting in a big speedup ($\times 7.4$ in our experiments, even more for bigger filters)

# Thank you for your attention!

# Questions?

Contact me at `charles.dubout@idiap.ch`